CITY UNIVERSITY LONDON

Safety and Assurance Cases introduction

DHS SW Assurance Forum, Sept 2010

Robin E Bloomfield

reb@adelard.com
reb@csr.city.sc.uk
College Building, City University, London EC1V 0HB
Tel: +44 20 7490 9450 (sec Adelard)
Tel: +44 20 7040 8420 (sec CSR)

CSR Building confidence in a computerised world
www.csr.city.ac.uk

Adelard

Overview

- Introduction
- Safety and assurance practices
- Supply chain experience
  - nuclear smart devices
  - financial system
- Extending to SCRM
- Threats and opportunities
- Conclusions and discussions

Adelard

- Safety and assurance cases and safety management systems
- Independent safety assessment
- Software assurance, including formal methods and static analysis
- Development, interpretation and application of standards and guidelines
- applied research in safety, security, critical infrastructure interdependencies
- policy to technology
- ASCE – the Assurance and Safety Case Environment
- clients in nuclear, defence, financial, transport sectors

Centre for Software Reliability

- Evaluation of socio-technical systems
  - Technical, interdisciplinary
- Research
  - with international community and users
- Education
  - placements, internships, scholarships, courses, MSc and CPD
- Innovation
  - director, Dr Peter Popov
  - DivSQL, PIA-FARA

CSR Building confidence in a computerised world
www.csr.city.ac.uk

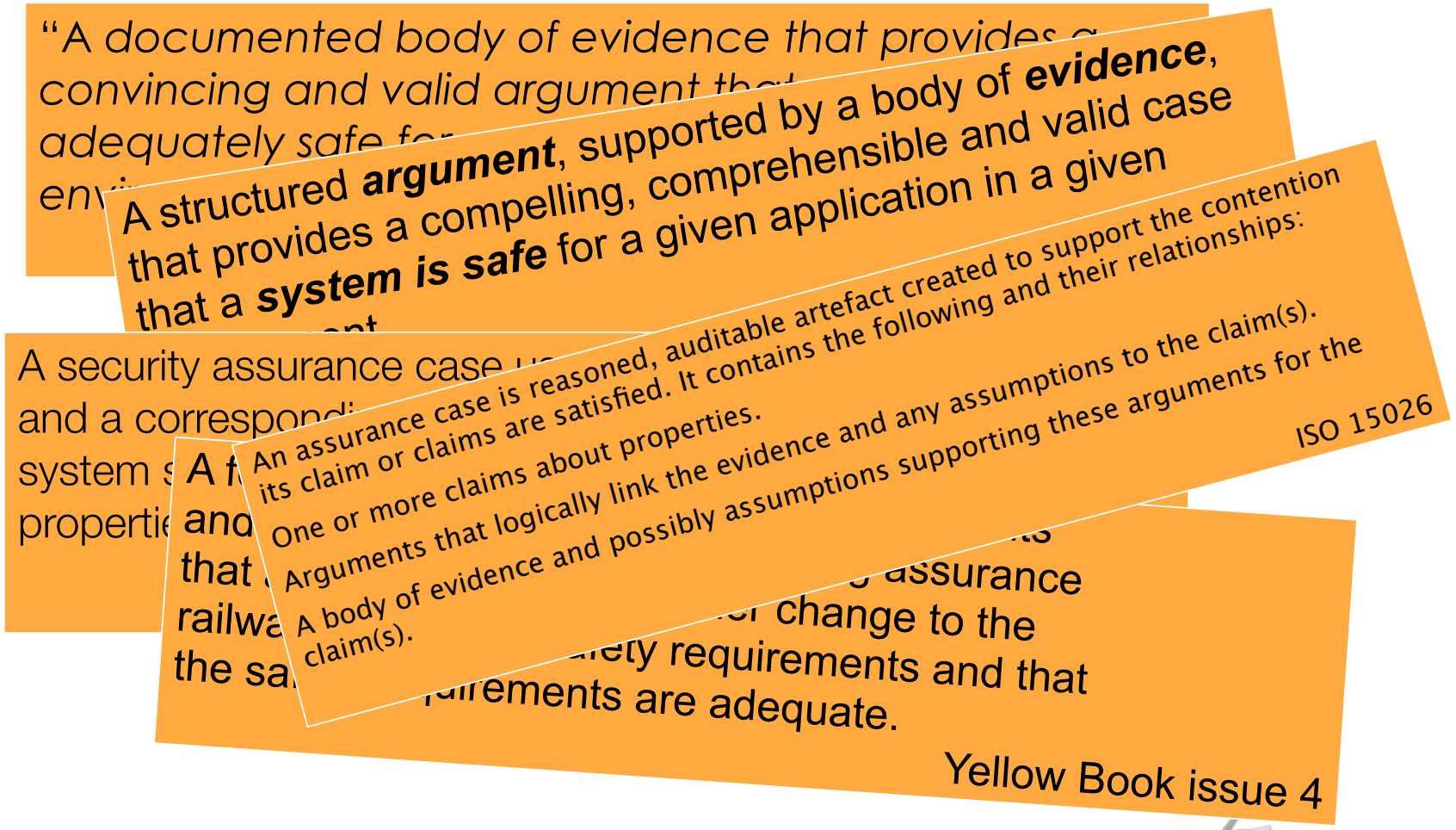
Adelard

In the beginning…

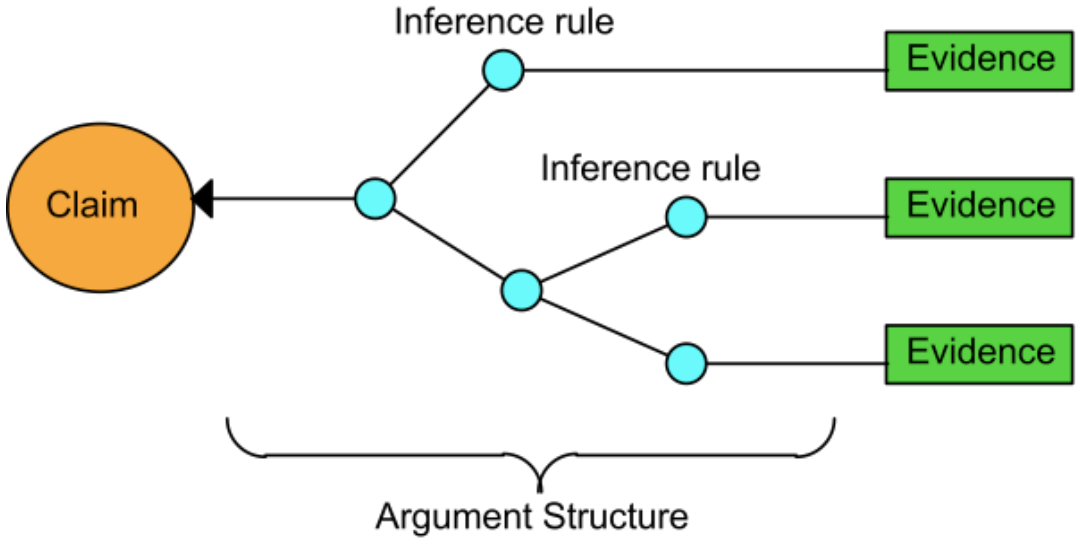
- “The World, according to the best geographers, is divided into Europe, Asia, Africa, America, and Romney Marsh",

wrote the Reverend Richard Harris Barham, writing as Thomas Ingoldsby, in the 1840s.

Some Definitions



Safety cases



- "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment"

Elements of a “Case”

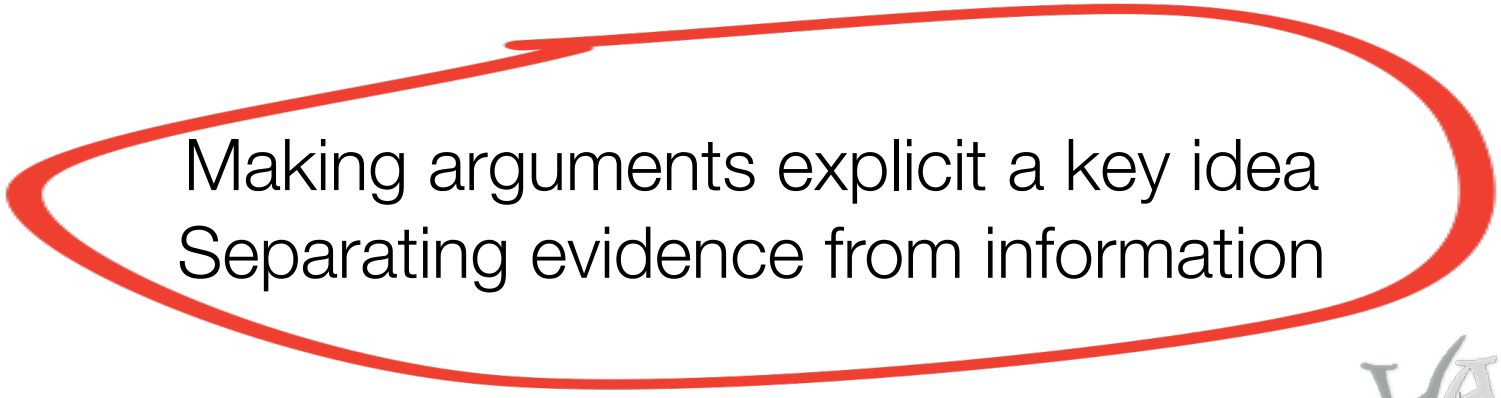
- Claim about a property of the system or some subsystem, with some confidence.
- Evidence that used as the basis of the trust argument. This can be either facts (e.g. based on established scientific principles and prior research), assumptions, or sub-claims, derived from a lower-level sub-argument.
- Argument linking the evidence to the claim, which can be deterministic, probabilistic or qualitative.

Types of argument

Deterministic or analytical application of predetermined rules to derive a true/false claim (given some initial assumptions), e.g. formal proof (compliance to specification, safety property), execution time analysis, exhaustive test, single fault criterion

Probabilistic quantitative statistical reasoning, to establish a numerical level, e.g. MTTF, MTTR, reliability testing

Qualitative compliance with rules that may have an indirect link the desired attributes, e.g. compliance with QMS and safety standards, staff skills and experience

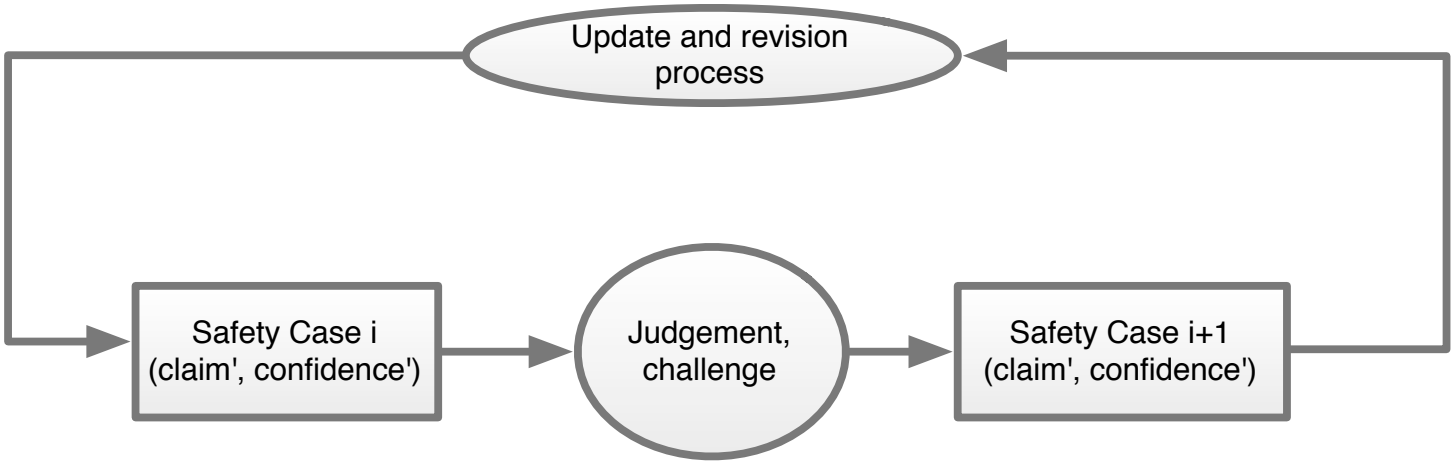


Communication and reasoning

- Structured safety and assurance cases have two essential roles:
  - communication is an essential function of the case, from this we can build confidence
    - boundary objects that record the shared understanding between the different stakeholders
  - a method for reasoning about dependability (safety, security, reliability, resilience ...) properties of the system
- Both are required to have systems that are trusted and trustworthy

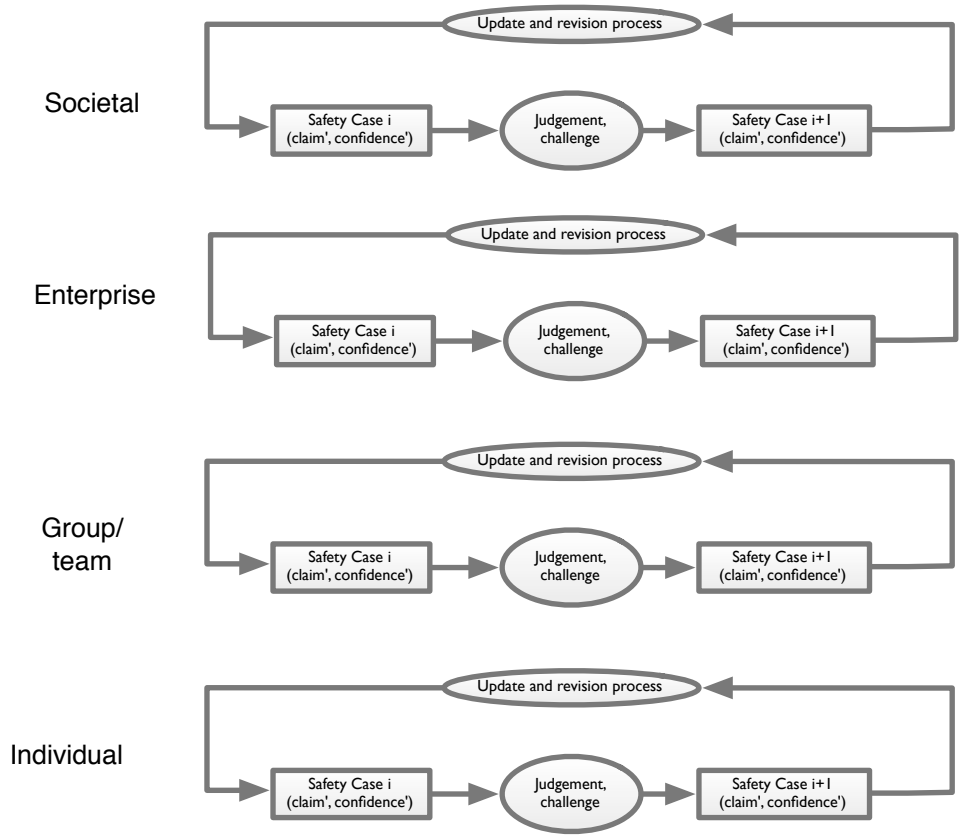
Safety case process – building confidence, challenging assumptions

- Captured in safety management system and in meta-case
- Challenge and response cycle essential
- Proof as a social, technical, adversarial process

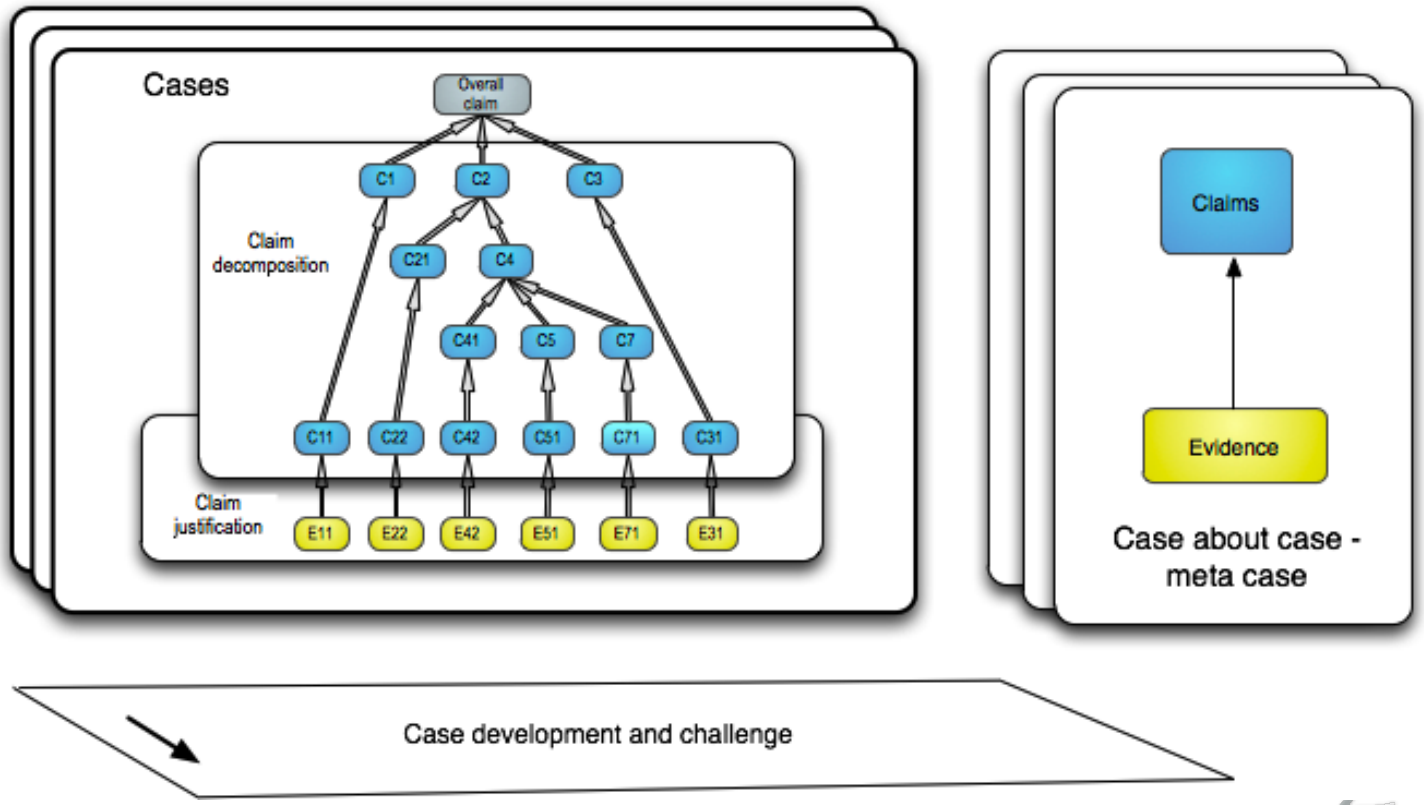


Safety case process – building confidence, challenging assumptions

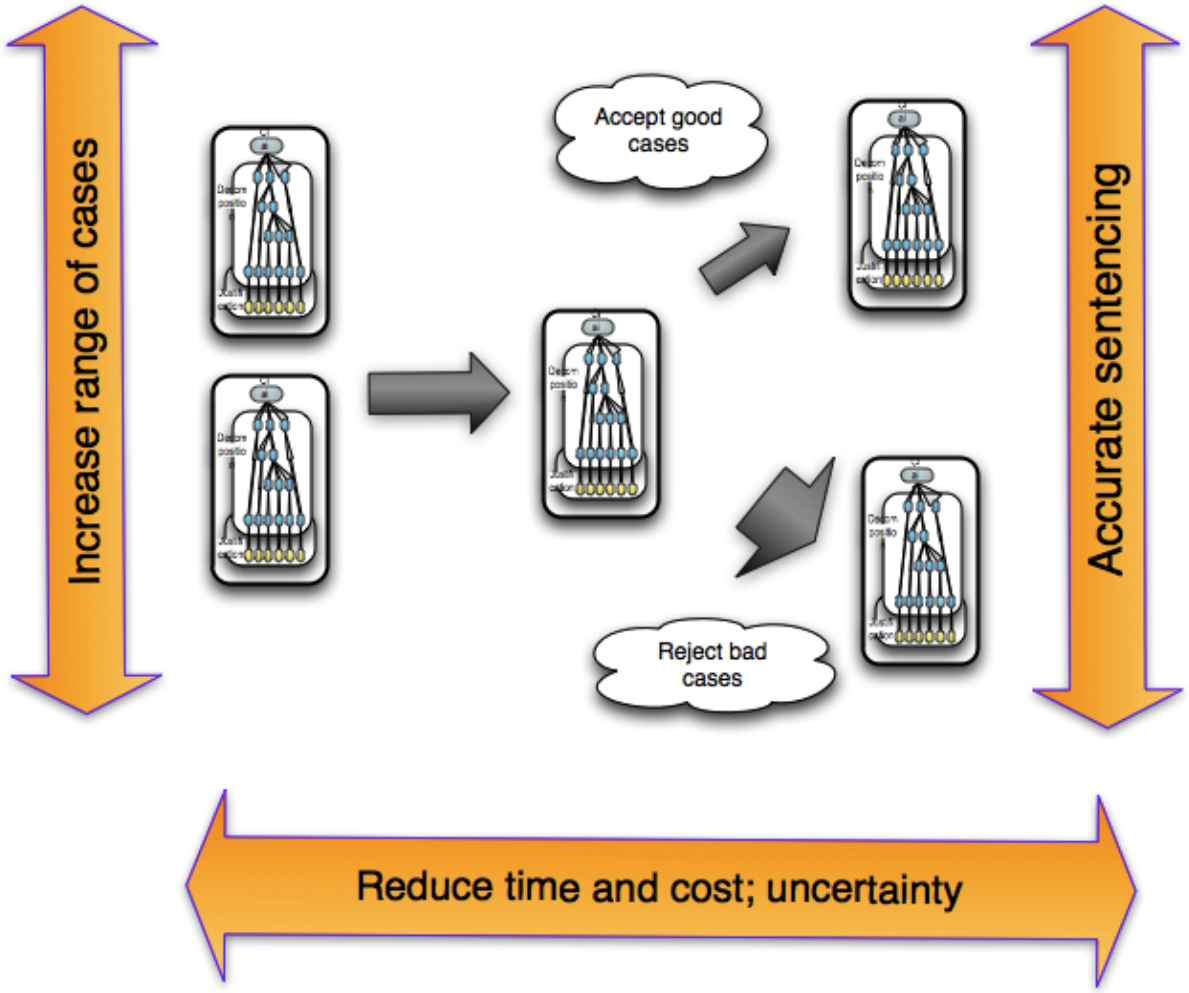
- Captured in safety management system and in meta-case
- Challenge and response cycle essential
- Proof as a social, technical, adversarial process



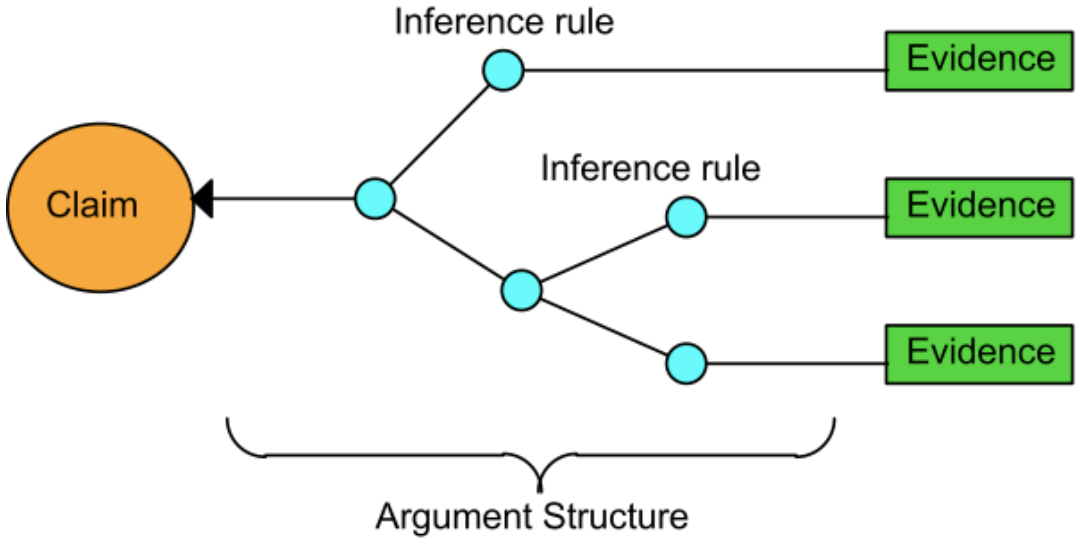
Reasoning, communication, confidence



Objectives

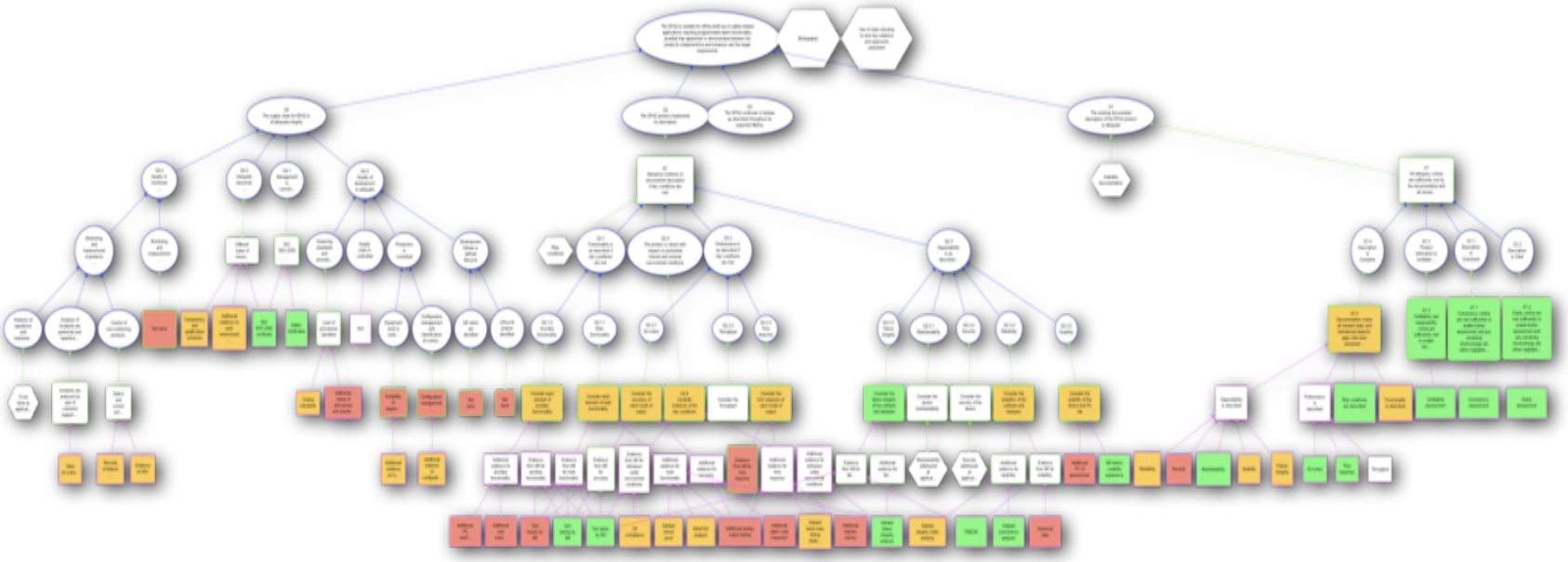

Increase range of cases
Accept good cases
Reject bad cases
Accurate sentencing
Reduce time and cost; uncertainty

In theory ...

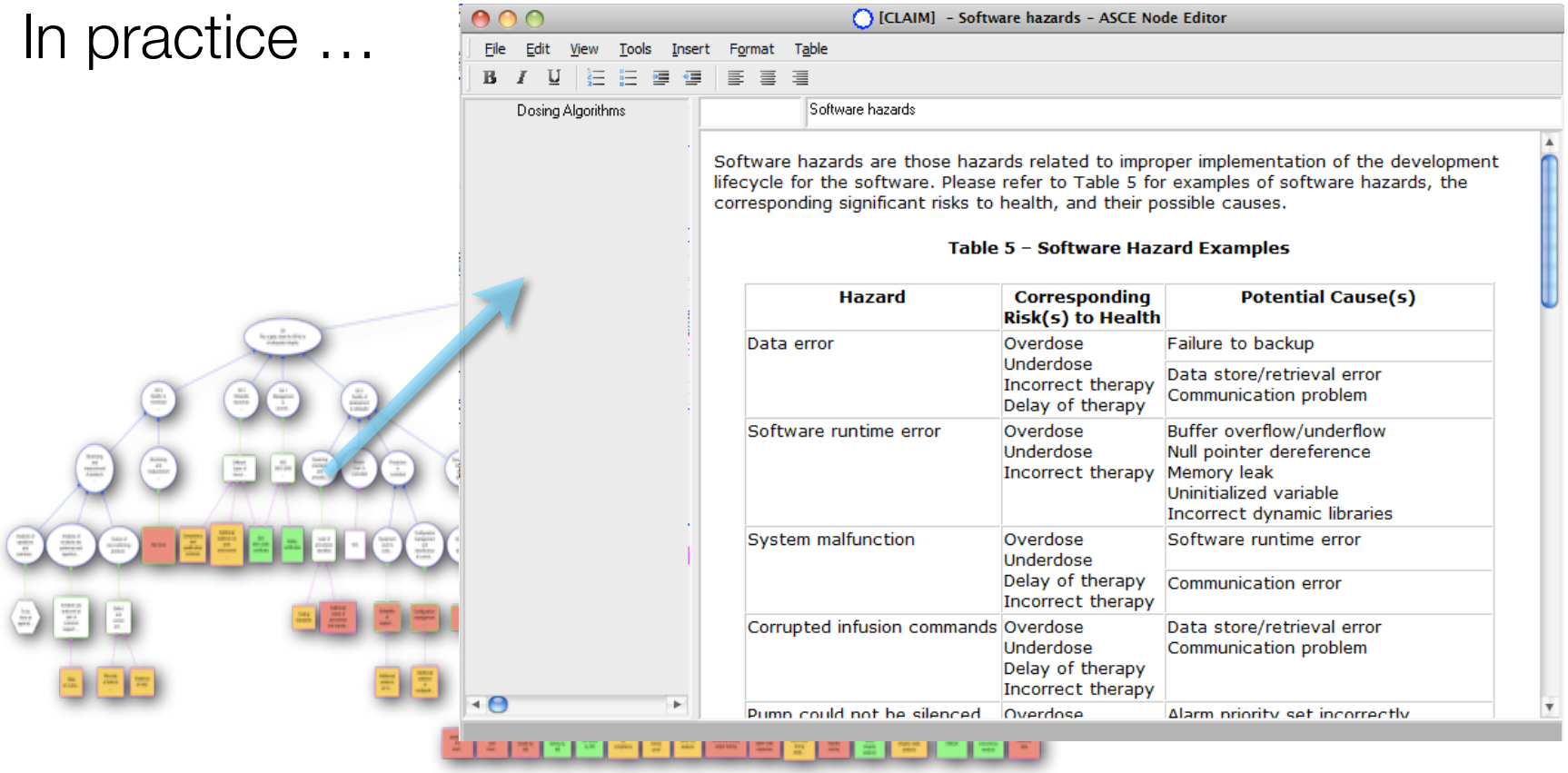


- "a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment"

In practice …



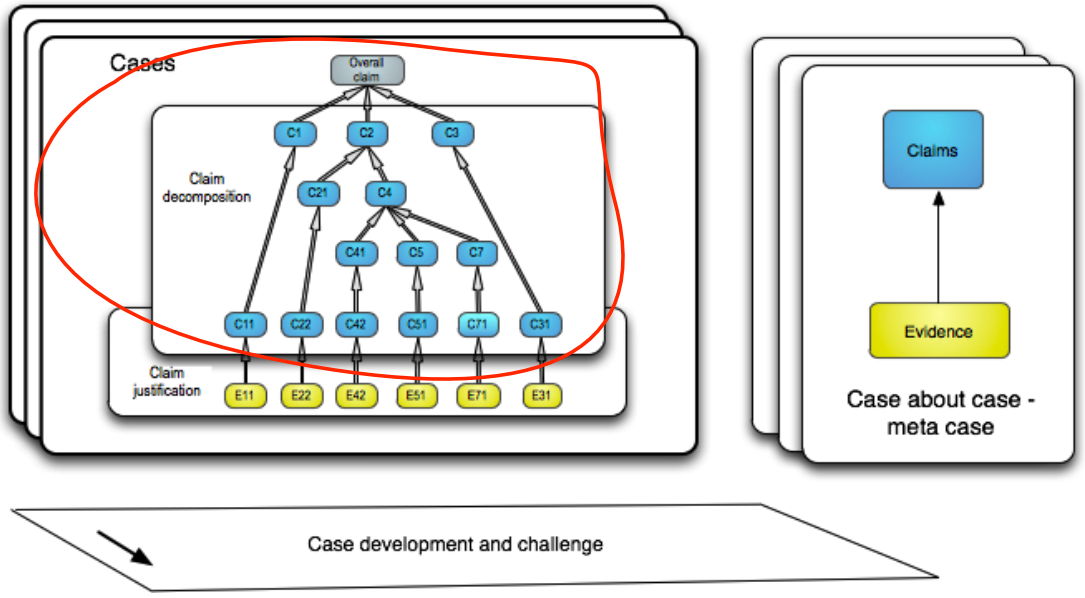
In practice …


[CLAIM] – Software hazards – ASCE Node Editor
File Edit View Tools Insert Format Table
Dosing Algorithms
Software hazards
Software hazards are those hazards related to improper implementation of the development lifecycle for the software. Please refer to Table 5 for examples of software hazards, the corresponding significant risks to health, and their possible causes.
Table 5 – Software Hazard Examples
Hazard
Corresponding Risk(s) to Health
Potential Cause(s)
Data error
Overdose
Underdose
Incorrect therapy
Delay of therapy
Failure to backup
Data store/retrieval error
Communication problem
Software runtime error
Overdose
Underdose
Incorrect therapy
Buffer overflow/underflow
Null pointer dereference
Memory leak
Uninitialized variable
Incorrect dynamic libraries
System malfunction
Overdose
Underdose
Delay of therapy
Incorrect therapy
Software runtime error
Communication error
Corrupted infusion commands
Overdose
Underdose
Delay of therapy
Incorrect therapy
Data store/retrieval error
Communication problem
Pump could not be silenced
Overdose
Alarm priority set incorrectly

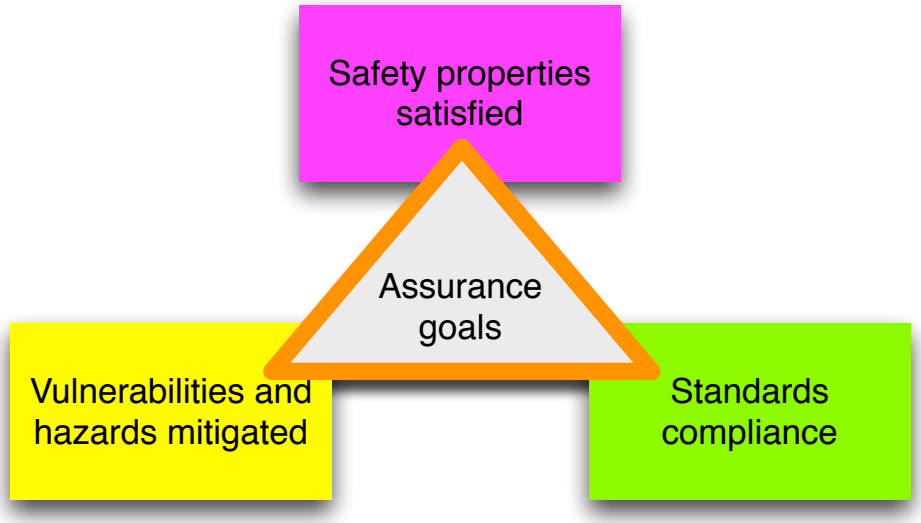
Architecting claim structure

Claim structure

- creative strategies
- claims language
- templates



Approaches


Safety properties satisfied
Assurance goals
Vulnerabilities and hazards mitigated
Standards compliance

Cases - argument styles

We have done what we were told to do (a *standards compliance* argument)

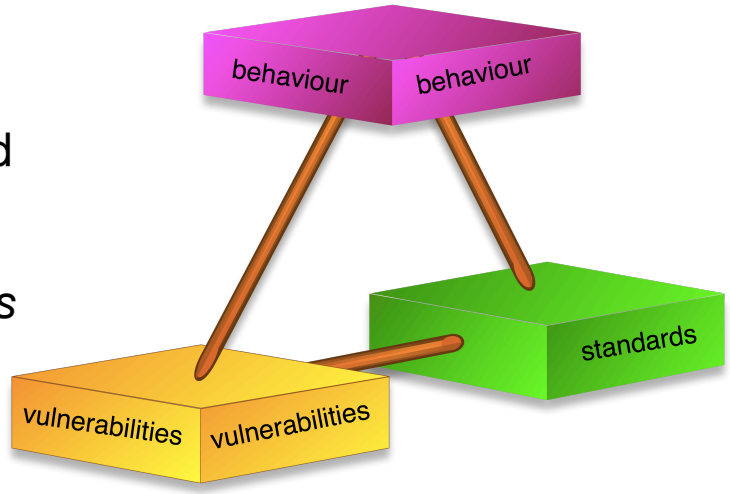
The system achieves the behaviour required (*safety properties* satisfied)

The system does not do bad things (*hazards addressed, vulnerabilities mitigated*)

Also

We have tried very hard (a *process argument*) to achieve dependability

Often a mixture of styles will be incorporated into a single case.

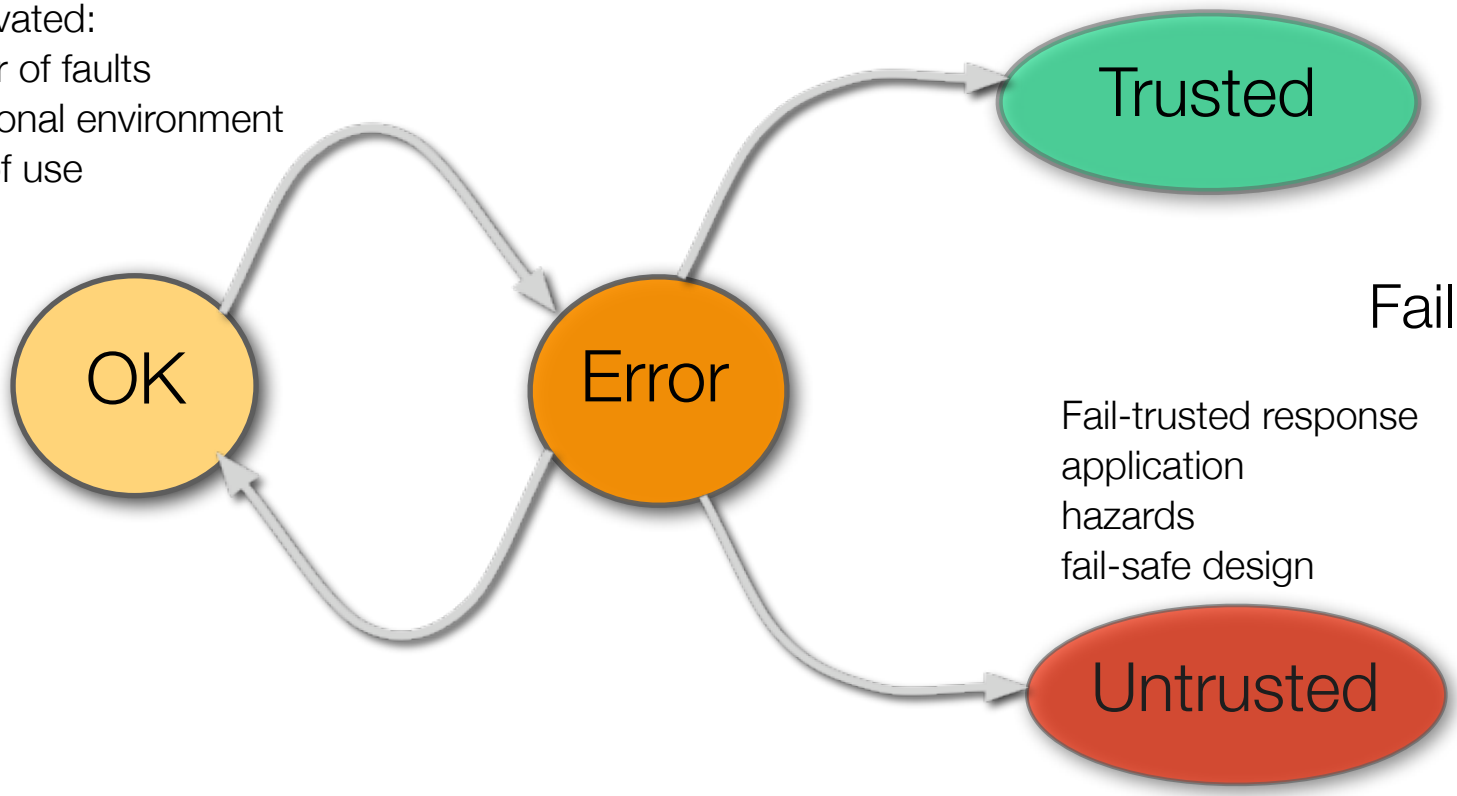


Standards and regulations

- Important part of case
- Can play different roles
  - Which needs to be justified
- But issues of validation
  - process -> product
  - techniques -> SIL achieved
- Need to innovate
  - Technology development V&V moves on
  - Use of COTS products
  - Product lines
  - Compliance can be expensive

Assurance strategies - behaviour

Fault activated:
- Number of faults
- Operational environment
- Mode of use



Failures

Fail-trusted response
application
hazards
fail-safe design

fault tolerance in design nature of application --
self healing, grace time

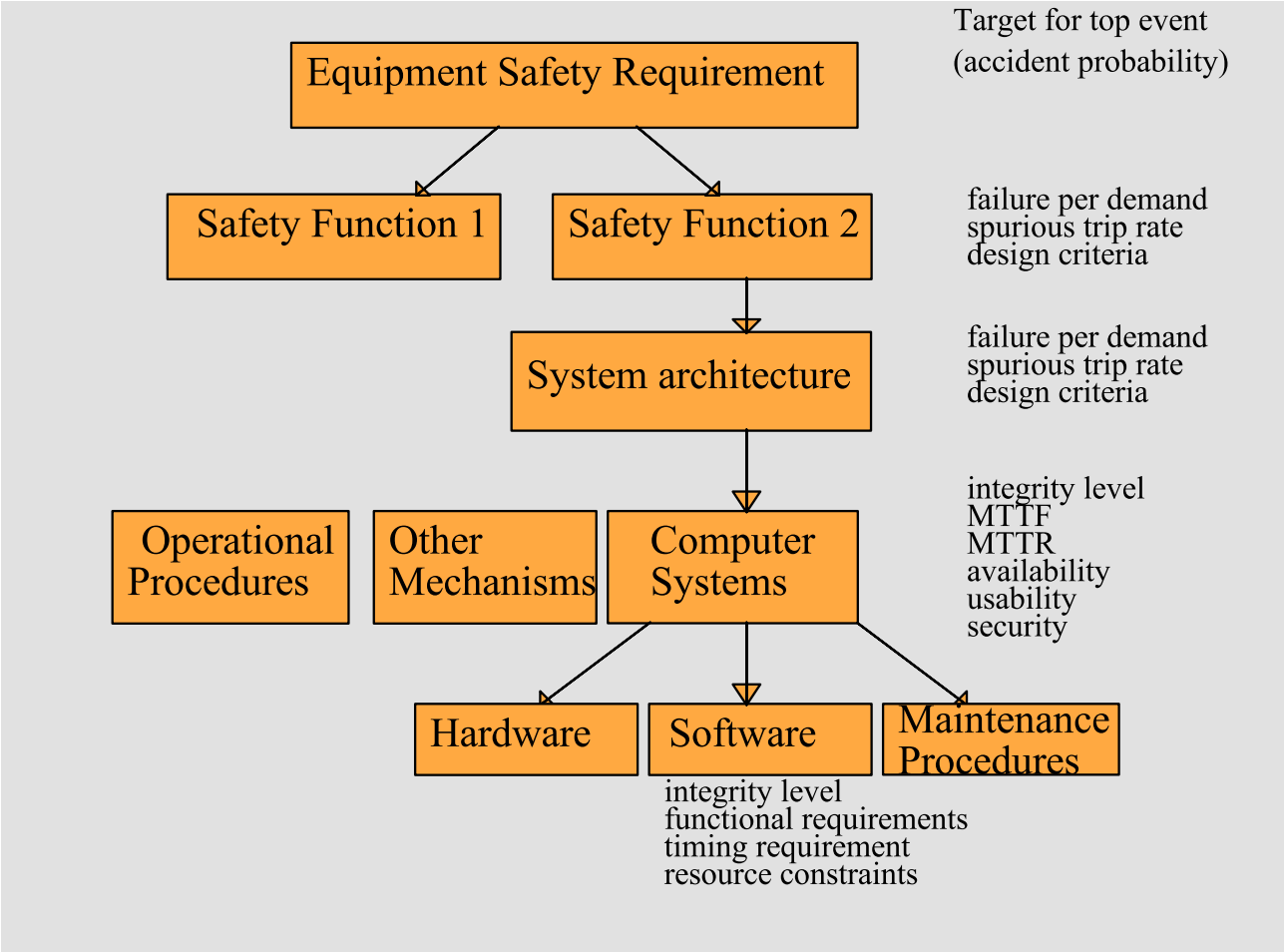
Strategies on behaviour

- Strategy – N No critical/significant fault or unsafe feature exists (the beast has no teeth, claws)
- Strategy –W Wrapper/containment argument – no failure or feature of the component can lead to hazard (the beast is in the cage)
- Strategy –R Restoration argument – any failure can be detected and recovered from (the beast can always be put back in the cage)
- And probabilistic variants of these

Safety properties and claims

- System safety analysis identifies hazards; these are amalgamated and abstracted into safety properties.
- Safety properties can be functions (shut down when T> 500), invariants (min sep always >2 miles) or purely descriptive (competency and culture).
- For each safety property address all attributes to increase completeness.
- As the design progresses need to consider derived properties arising from hazards introduced by the implementation.
- Non-functional system properties evolve
- May be claim limits

Architecture and functional claim expansion



Claim attribute expansion

- Claims can be broken down into claims about different attributes for the various sub-systems, e.g.:

reliability and availability
usability (by the operator)
security (external attack)
fail-safe response
functional correctness

accuracy
time response
robustness to overload
maintainability
modifiability, etc.

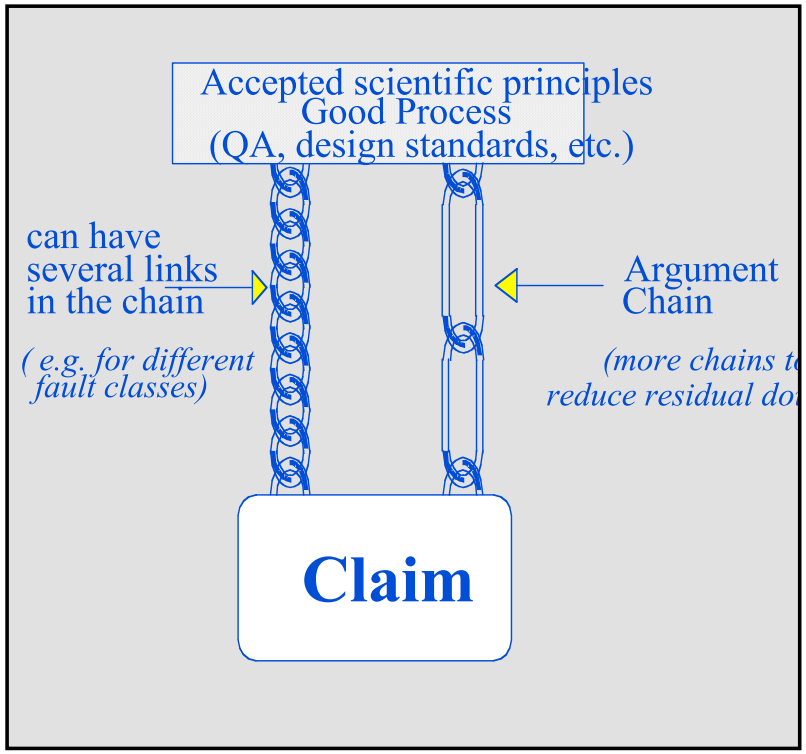
Restricted types of claim expansion

- Claim expansion language initially unconstrained
  - CAE
  - (also of course GSN)
- Empirically found a small set of constructs useful
- These enable more formal underpinnings and pragmatic checklists
- Uniformity and regularity in cases
- Allows us to asses cases
- Gradually introduced in our work

Main types – keywords	Comment
architecture	splitting a component into several others
functional	
property decomposition	splitting a property into several others e.g. set of attributes
infinite set	inductive partitioning (e.g., over time)
complete	capturing the full set of values for risks, requirements, etc.
monotonic	the new system only improves on the old system
concretion	making informal statements less vague
generalises	property shown for one member of a class and generalised to all others
an-instance-of	properties shown for all components of a certain class

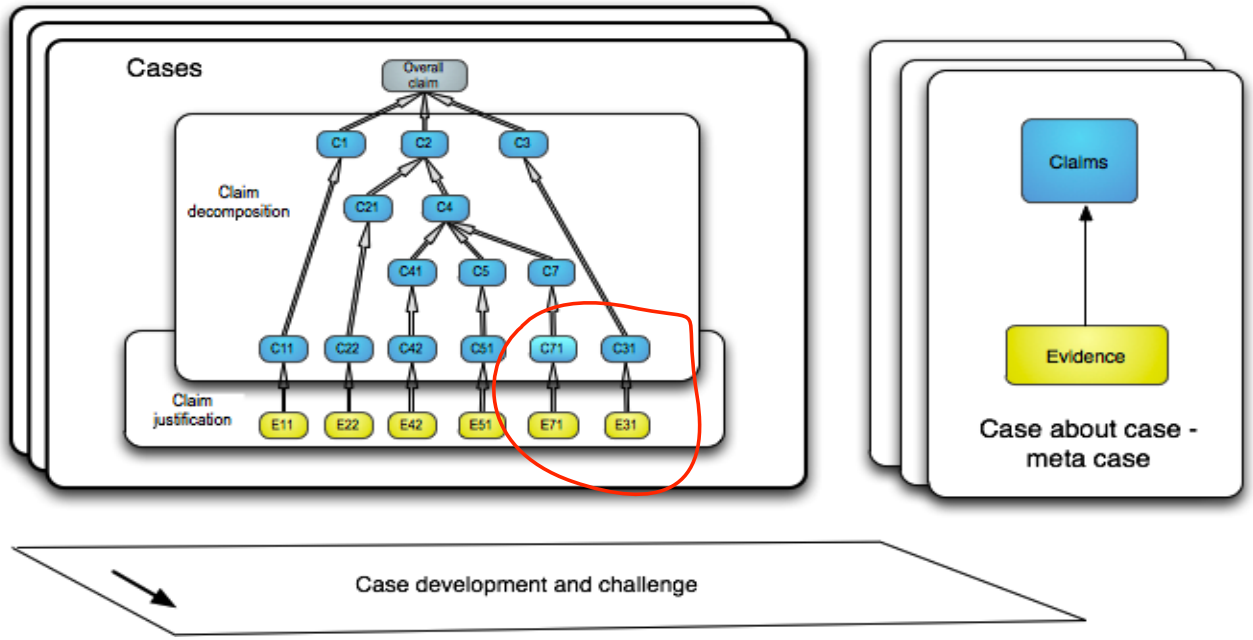
Argument metaphors

- Architecture of cases
- There is a parallel between architecture and argument structure
- e.g. in use of diversity, single failure criterion, sensitivity studies
- metaphors of “belt and braces”, “legs to stand on”
- formalisation difficult and current research topic



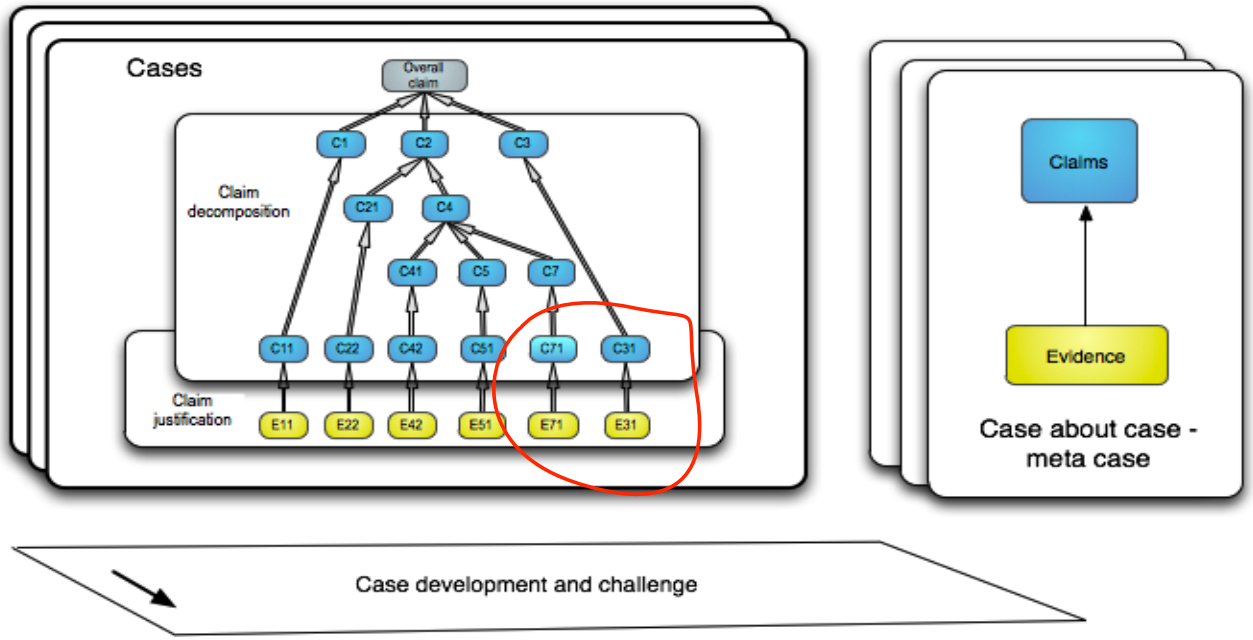
Map evidence to claims

- iterative selection of techniques that generate evidence



Map evidence to claims

- iterative selection of techniques that generate evidence



Selecting techniques and activities to generate evidence

- Catalogues of techniques e.g. in IEC 61508 Part3
  - P Bishop book
- Standards leave it as “exercise for the reader” in justifying selection
  - Supported by case
- Two useful mappings are
  - Activities/techniques → role in case
  - Attributes -> techniques
- Examples tables

Technique	Aim	Category	Assurance achieved	Effort	Expertise
Competence management	Assess competency management. Improve software quality by team with adequate competence.	FP	Indirect assurance from competence of development team.	Some additional management overheads.	Low, although assessment of requirements needs domain knowledge
Review of requirements process	Assess requirements process and requirements traceability.	FP	Increase confidence in requirements validity and satisfaction.	Information gathering may take a long time, depending on the complexity of the system.	High, as it needs to focus on what it is important. Need understanding of the system, vulnerabilities, weaknesses in both documents, process and specification
Review of quality of supply					
Supplier competency	Improve software quality by team with adequate competence.	FP	Indirect assurance from quality of development process.	Low	Low.

Acknowledgments

- Colleagues in CSR and Adelard, particularly Peter Bishop, George Cleland, Lukasz Cyra, Sofia Guerra, Dan Sheridan, Bev Littlewood, Andrey Povyakalo, Lorenzo Strigini and others